

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-338521

(43) 公開日 平成11年(1999)12月10日

(51) Int.Cl.⁶

識別記号

F I

G 0 5 B 19/05

G 0 5 B 19/05

A

B 2 5 J 9/16

B 2 5 J 9/16

G 0 6 F 9/06

5 3 0

G 0 6 F 9/06

5 3 0 C

9/45

9/44

3 2 0 C

審査請求 未請求 請求項の数 1 O L (全 12 頁)

(21) 出願番号

特願平10-147267

(22) 出願日

平成10年(1998) 5 月28日

(71) 出願人 000006208

三菱重工業株式会社

東京都千代田区丸の内二丁目5番1号

(72) 発明者 石井 圭

兵庫県高砂市荒井町新浜2丁目1番1号

三菱重工業株式会社高砂研究所内

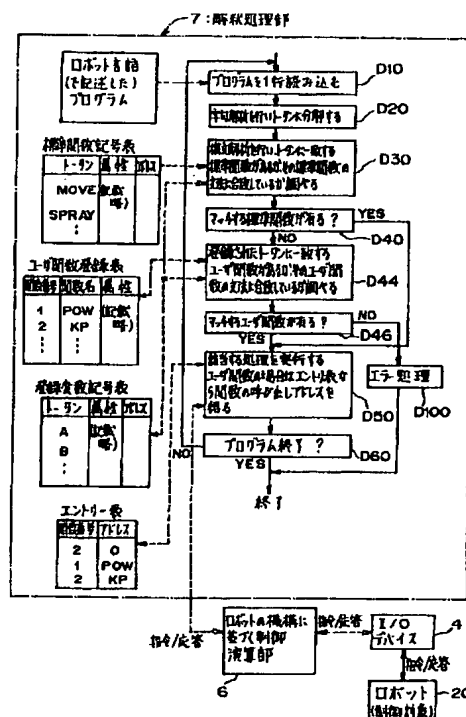
(74) 代理人 弁理士 真田 有

(54) 【発明の名称】 制御装置

(57) 【要約】

【課題】 汎用ロボットやプラントの制御に用いて好適の制御装置において、システム自体の改造をすることなく、かつ、制御対象に複雑な作業を行なわせる場合にも作業効率を低下させることなく、ユーザが必要な動作を制御対象にさせることができるようにする。

【解決手段】 ロボット言語又は図形言語により記述された制御プログラムに対して標準関数を用いて言語解釈及び演算処理を行なう解釈処理部7と、解釈処理部7による該言語解釈及び該演算処理によって得られた指令情報に従って制御対象20に動作指令を行なう制御演算部6とをそなえた制御装置において、ユーザが該標準関数とは別のユーザ関数を登録するユーザ関数登録機能をそなえ、解釈処理部7は、該ユーザ関数に対応した該制御プログラムに対しては該ユーザ関数を用いて言語解釈及び演算処理を行なうように構成する。



【特許請求の範囲】

【請求項 1】 ロボット言語又は図形言語により記述された制御プログラムに対して標準関数を用いて言語解釈及び演算処理を行なう解釈処理部と、該解釈処理部による該言語解釈及び該演算処理によって得られた指令情報に従って制御対象に動作指令を行なう制御演算部とをそなえた制御装置において、ユーザが該標準関数とは別のユーザ関数を登録するユーザ関数登録機能をそなえ、該解釈処理部は、該ユーザ関数に対応した該制御プログラムに対しては該ユーザ関数を用いて言語解釈及び演算処理を行なうことを特徴とする、制御装置。

【発明の詳細な説明】**【0001】**

【発明の属する技術分野】 本発明は、ロボット言語により制御可能な汎用ロボットの制御、あるいは、図形言語によりプラント制御ロジックを記述して制御可能なプラントの制御に用いて好適の、制御装置に関する。

【0002】

【従来の技術】 従来より、ロボットの制御やプラントの制御を行なう制御装置が開発されている。このような従来の制御装置では、ユーザが容易に制御プログラムを記述して用途に応じた動作指令を制御対象に行なうことができるように、標準関数をそなえた基本ソフトウェアが用意されている。ユーザは、この標準関数を組み合わせ、目的に応じた制御プログラムを記述している。特に比較的に緻密な作業に使用されることの多いロボットでは、適宜に動作の修正又は変更を要求されることが多いため、ユーザが目的に応じてロボットに動作指令を行なわせることを容易にできる制御装置の要求は高い。

【0003】 ロボットとしては、例えば、塗装スプレーを行なう塗装ロボットがある。図 7～図 10 は従来の制御装置にかかる塗装ロボットを示す図である。図 7 は塗装ロボットの構成及び塗装ロボットの動作内容を示す図であり、図 8 はロボット言語（標準関数）による塗装ロボットの制御例を示す図であって、(a)、(b) はそれぞれ図 7 の動作を塗装ロボットに行なわせるためのロボット言語による制御プログラムを示す図であり、図 9 はロボット制御装置の内部構成を示す図であり、図 10 はロボット言語により記述した制御プログラムの解釈処理のフローを説明する装置の要部構成図である。

【0004】 図 7 に示すように、塗装ロボット 20 は、塗装スプレー 21 と、塗装スプレー 21 を移動させる移動機構 25 と、塗装スプレー 21 及び移動機構 25 に動作指令を行なうロボット制御装置 10 とから構成され、作業対象 40 にスプレー塗装を行なう。ここで、移動機構 25 は、軸 26a、27a、28a、29a と、これらの軸 26a～29a の軸間及び軸端に設けられた傾動関節 26、27、29 及び回動関節 28、30 とをそなえている。傾動関節 26、27、29 は、結合された軸

を傾動できるようになっており、回動関節 28 は、接続された軸をその軸心線回りに回動できるようになっている。回動関節 30 は、塗装ロボット 20 を支持しており、接続された軸をその軸心線回りに片方向に回転できるようにになっている。

【0005】 これらの傾動関節 26、27、29、回動関節 28、30 には、ロボット制御装置 10 からの動作指令で動作する図示されない関節駆動装置、例えば、モータ及び減速機がそれぞれ設置されている。また、傾動関節 26、27、29、回動関節 28、30 には、エンコーダがそれぞれ付設されており、これらのエンコーダにより検出した傾動関節 26、27、29、回動関節 28、30 の動作量をロボット制御装置 10 にフィードバックしている。

【0006】 そして、ロボット制御装置 10 の内部は、図 9 に示すように、CPU1 と、RAM2 と、ROM3 と、I/O デバイス 4 と、バッテリーバックアップ付きの RAM5 と、それらを互いに繋げる伝送路 9 とから構成される。

【0007】 ここで、塗装ロボット 20 を制御するための基本ソフトウェアは ROM3 に格納されており、CPU1 はこの基本ソフトウェアを RAM2 を使って実行する。制御プログラムは、図 8 に示すようなロボット言語により記述されており、電源を切っても内容が失われないメモリ領域、ここでは、バッテリーバックアップ付きの RAM5 に通常ロードされている。このバッテリーバックアップ付き RAM5 の代わりに、フラッシュ ROM などを使うこともある。そして、ロボット制御装置 10 は I/O デバイス 4 を介して塗装ロボット 20 とつながれている。

【0008】 塗装ロボット 20 は上述のように構成されるので、図 8 (a)、(b) に示すようなロボット言語（標準関数）による制御プログラムを作成してロボット制御装置 10 へ入力することで、塗装ロボット 20 に作業対象物 40 の所定の位置に塗装を行なわせることができる。この制御プログラムは、ロボット制御装置 10 の CPU1 で解釈処理された後に、I/O デバイス 4 を介して、動作信号として塗装スプレー 21 及び移動機構 25 の上述の関節駆動装置に送られる。この動作信号に応じて、塗装スプレー 21 は塗装物の噴射の開始又は停止の何れかを行ない、各関節駆動装置は互いに連動して塗装スプレー 21 を所定の位置へと移動させる。

【0009】 ここで、図 8 (a) のロボット言語により記述された制御プログラムについて説明する。この記述内容は、塗装ロボット 20 により、図 7 に示す作業対象物 40 の、座標 1、2 間及び座標 3、4 間の範囲で塗装を行なわせるためのものである。ここで、座標 1～4 は何れも作業対象物 40 の端点に位置する。以下に、図 8 (a) の制御プログラムを、図 7 と対応させながら、上から一行ずつ説明していく。

【0010】「MOVE」は、塗装スプレー 21 を、引数で指定した座標へ移動させるための関数である。ここでは、「MOVE 1」として引数に 1 をセットして、塗装スプレー 21 を塗装の開始点で作業対象物 40 の端点に位置する座標 1 へと移動させる（ステップ A 1）。次の「OVERSPRAY」は引数で指定した量をオーバー量に設定する関数である。ここで、オーバー量とは、塗装スプレー 21 の、塗装物の噴射停止時における不安定な噴射状態による塗装むらを考慮して設定するもので、OVERSPRAY 以降に記述される MOVE については、その引数よりも OVERSPRAY により設定したオーバー量だけ離れた座標へと塗装スプレー 21 を移動させる。さらに、OVERSPRAY 以前に MOVE 指令により塗装スプレー 21 を移動させている場合は、まず、この MOVE 指令による移動とは逆方向にオーバー量だけ塗装スプレー 21 を引き戻す。

【0011】ここでは、「OVERSPRAY 50」としてオーバー量を 50 mm に設定している。したがって、塗装スプレー 21 は、直前の MOVE（ステップ A 1）による移動に対して、OVERSPRAY により設定した量（オーバー量）だけ手前の図中 A の位置へと引き戻される。（ステップ A 2）。

【0012】塗装スプレー 21 が図中 A の位置へ移動した時点で、次の「SPRAY ON」により、塗装スプレー 21 の塗装物の噴射を開始させる（ステップ A 3）。次に、「MOVE 2」として引数に 2 をセットして、塗装スプレー 21 を、MOVE 関数により、作業対象物 40 の端点に位置する座標 2 へ移動させるように指令しているが、ステップ A 2 の OVERSPRAY の設定により、MOVE 関数により指定された位置（座標 2）に対し OVERSPRAY により設定した量（オーバー量）だけ先の位置（図中 B の位置）へと移動させる（ステップ A 4）。

【0013】塗装スプレー 21 が図中 B の位置へ移動した時点で、次の「SPRAY OFF」により、塗装スプレー 21 の塗装物の噴射を停止させる（ステップ A 5）。次の「MOVE 3」は、塗装スプレー 21 を、MOVE 関数により、座標 3 へ移動させるように指令しているが、MOVE 関数により指定された位置（座標 3）に対し、OVERSPRAY で設定したオーバー量だけ手前の位置（図中 C の位置）へと移動させる（ステップ A 6）。

【0014】塗装スプレー 21 が図中 C の位置へ移動した時点で、次の「SPRAY ON」により、塗装スプレー 21 の塗装物の噴射を開始させる（ステップ A 7）。次の「MOVE 4」は、塗装スプレー 21 を、MOVE 関数により、座標 4 へ移動させるように指令しているが、MOVE 関数により指定された位置（座標 4）に対し、OVERSPRAY で設定したオーバー量だけ先の位置（図中 D の位置）へと移動させる（ステップ

A 8）。

【0015】塗装スプレー 21 が図中 D の位置へ移動した時点で、次の「SPRAY OFF」により、塗装スプレー 21 の塗装物の噴射を停止させ（ステップ A 9）、塗装ロボット 20 の作業が終了する。この図 8（a）に示す制御プログラムにより、ロボット制御装置 10 より動作指令を受けた塗装ロボット 20 は、図 7 に示す作業対象物 40 に対して、以下のように動作する。

【0016】塗装スプレー 21 は、まず、座標 1 の手前のある点から、作業対象物 40 の端点に位置して塗装開始点である座標 1 の上方へと移動する（ステップ A 1）が、その後、図中の A で示される位置まで後退して（ステップ A 2）、この A の位置で塗装物の噴射を開始する（ステップ A 3）。そして、塗装スプレー 21 は、塗装物を噴射しながら、移動機構 25 により、この A の位置から、座標 1 及び所定の座標 2 を通過して図中の B で示される位置まで移動して（ステップ A 4）、塗装物の噴射を停止する（ステップ A 5）。塗装開始点（座標 1）よりもの所定量（50 mm）手前の地点 A から塗装物の噴射を開始したり、塗装物の噴射停止点（座標 2）を所定量（50 mm）オーバーした地点 B まで塗装物を噴射させるのは、塗装スプレー 21 が停止した状態で、塗装物の噴射開始または停止を行なうと、不安定な噴射状態による塗装むらが塗装物の噴射開始点や停止点（座標 1, 2）付近で生じるおそれがあるので、これを防止するためである。

【0017】次に、塗装スプレー 21 は、移動機構 25 により図中 C の位置に移動し（ステップ A 6）て、この C の位置で塗装物の噴射を開始し（ステップ A 7）、塗装物を噴射しながら移動機構 25 により図中の D で示される位置まで移動して（ステップ A 8）、塗装物の噴射を停止する（ステップ A 9）。

【0018】塗装を行なう所定範囲（座標 3, 4 間）よりも、手前の位置（座標 C）から塗装物の噴射を開始して、先の位置（座標 D）まで塗装物を噴射させるのは、上述と同様に、塗装物の噴射開始点及び噴射停止点（座標 3, 4）付近での不安定な噴射状態による塗装むらが生じるのを防ぐためである。以上で、塗装ロボット 20 の作業は終了する。

【0019】ここで、図 8（a）の制御プログラムを、図 8（b）の制御プログラムのように、即ち、図 8（a）の制御プログラムに対し、ステップ A 1 の MOVE 1 とステップ A 2 の OVERSPRAY 50 の順序を逆にして記述することもある。この場合、図 8

（a）の制御プログラムのステップ A 1 からステップ A 3 にかかるスプレーロボット 20 の動作は、前述の通り、図 7 において、塗装スプレー 21 が、座標 A を通りすぎて、塗装開始点である座標 1 の上方へと一旦移動した後に座標 A の位置まで引き返して塗装物の噴射を開始するのにに対し、図 8（b）の制御プログラムのステップ

E1からステップE3にかかるスプレーロボット20の動作は、MOVE1以前にOVERSPRAYによりオーバー量が設定されているので、塗装スプレー21は、座標1まで移動してからAの地点まで引き返すといった動作なしに、直接に、塗装開始点である座標1よりもオーバー量だけ手前の位置である座標Aに移動し、そして塗装物の噴射を開始する。

【0020】図8(b)の制御プログラムによる動作は、座標Aでの塗装物の噴射を開始以降については、図8(a)の制御プログラムによる動作と同じであるので説明を省略する。

【0021】次に、ロボット言語の解釈処理について説明する。ロボット言語の解釈処理は、基本ソフトウェアにより、例えば、図10に示すように行なわれる。ここで、解釈処理部7は、言語解釈(ステップB10～ステップB40)と演算処理(ステップB50)とを行ない、言語解釈及び演算処理後に指令を制御演算部6に出力する。制御演算部6は、この指令によってさらに指令値を計算してI/Oデバイス4を介して塗装ロボット20へ動作指令を出力する。また、塗装ロボット20からは、塗装ロボット20の位置等のロボットの状態を示す信号がロボット制御装置10にフィードバックされる。

【0022】まず、基本ソフトウェアによって、図8(a)、(b)に示すような制御プログラムを1行ずつ読み込み(ステップB10)、そして、読み込んだプログラム行を字句解析を行なって最小単位の語句(トークン)に分解する(ステップB20)。ここで、制御プログラムは、基本ソフトウェアで解釈処理しやすいように適当な中間言語にコンパイルされてロードされている場合もある。

【0023】次いで、このトークンを、標準関数記号表及び登録変数記号表と比較して構文解析を行なう(ステップB30)。ここで、標準関数記号表には、その属性(関数の引数の数及び引数の型等)を含んで標準関数が登録されている。さらに、この標準関数記号表には標準関数を実際に処理するためのアドレスも登録されており、速やかに標準関数の処理の実行が行なえるようになっている。

【0024】また、登録変数記号表も、標準関数と同様の内容を持っている。つまり、登録変数記号表には、使用される変数が、変数の型等の属性を含んで登録され、さらに、変数の実際のアドレス等の情報も含まれており、変数の内容を速やかに検索できるようになっている。つまり、構文解析では、トークンと標準関数登録表及び登録変数記号表とを比較してトークンが標準関数と合致しているかを調査する。即ち、分解されたトークンが、上述の標準関数記号表に登録されているいずれかの標準関数と一致するとともにその標準関数の属性(標準関数の文法)に適っているか、さらに、標準関数の引数に変数が使われている場合は、その変数が登録変数記号

表に登録されている変数のいずれかと一致するとともにその変数の属性に適っているかを調査する。

【0025】以上で、構文解析が終了し、ステップB40へと進む。ステップB40では、トークンが標準関数に合致しているかの判定がおこなわれる。ここで、トークンが標準関数に合致していなければ、ステップB10へと進みエラー処理が行なわれて、制御プログラムの解釈処理は未完のまま強制終了となるが、トークンが標準関数に合致していれば、ステップB50へと進む。

【0026】ステップB50では、トークンと合致する標準関数に該当する処理を行なうように制御演算部6に指令を出してステップB60へと進む。ここで、ステップB50、即ち、解釈処理部7から、解釈処理の結果得られた動作命令や移動先の座標などの指令を受けると、制御演算部6は、さらに指令値を計算して、この指令値(動作指令)を、I/Oデバイス4経由でロボットの各アクチュエーター(塗装ロボット20)に出力する。

【0027】ステップB60では、ステップB10で読み込んだ制御プログラムの1行が制御プログラムの最終行かどうか、即ち、制御プログラムが終了かどうかの判定が行なわれ、制御プログラムが終了であれば解釈処理は終了するが、制御プログラムが終了でなければ、ステップB10へと戻って、制御プログラムの次の行を読み込み、制御プログラムが終了するまで解釈処理(ステップB10～ステップB60)を繰り返す。

【0028】以上で、基本ソフトウェアによるロボット言語の解釈処理のフローが終了する。このように、従来のロボット制御装置では、いくつかの簡単なロボット言語(標準関数)を組み合わせることで制御プログラムを記述することでロボットに複雑な動作を行なわせることができるようになっている。ロボット言語としては、例えば、汎用的なプログラム言語であるBASICに類似したものがJIS規格にも定められている。

【0029】このようなロボット言語の他に、プラント制御装置では、図11のようなファンクションブロック図などの図形形式の言語を使って制御プログラムを記述するものもある。図11の制御プログラムは、プラント制御装置にアナログ入力されている、図11中①で示す制御対象となるある状態量の制御目標値(Mokuhyo)と図11中②で示すこの状態量の現状の制御状態(Seigyoryo)との差分を、図11中③で示す減算関数により算出し、さらに、この差分を使って図11中④で示す関数 $\left[\frac{K}{1+Ts} \right]$ により一次遅れを考慮した操作量を算出して、この操作量を、制御対象となる状態量の調節機器に指令値としてアナログ出力させる(図11中⑤)ものである。

【0030】

【発明が解決しようとする課題】しかしながら、このような従来の制御装置では、必要な動作を制御対象に行なわせるために必要な制御プログラム(必要制御プログラ

ム)を記述するのにユーザが使用できるのは、制御装置に標準で提供されているロボット言語(標準関数)に限られているため、以下の課題がある。

【0031】制御プログラムにより例えば汎用ロボットのような制御対象に動作を行なわせるには、図10に示すように、制御プログラムを構成する標準関数のそれぞれについて解釈処理が必要であり、このため、1つの制御プログラムを実行するのに要する解釈処理時間は制御プログラムを構成する標準関数の数に比例して増加する。したがって、制御対象に複雑な動作を行なわせる必要があって必要制御プログラムを構成する標準関数が多数になるほど、解釈処理時間が増大し、ロボットやプラント等の制御対象の作業効率の低下を招いてしまう。

【0032】ここで、複数の標準関数をまとめて1つのマクロ関数としてユーザが定義可能な、マクロ機能を持つ制御装置もある。複数の標準関数からなる類似の動作を繰り返し制御対象に行なわせるときには、この複数の標準関数を1つのマクロ関数とすることによりユーザの制御プログラム入力作業の煩雑さを解消することができるが、マクロ関数を構成する複数の標準関数のそれぞれについて解釈処理を行なっているため、上述と同様で解釈処理時間を短くすることはできず、制御対象の作業効率の低下を緩和するには至らない。

【0033】さらに、制御装置に標準で提供される標準関数では、必要制御プログラムを記述しきることが困難な場合もある。このような場合、ユーザは、制御装置のシステム自体を高価な費用を払ってメーカに改造してもらるか、また、何とか既存の標準関数を組み合わせて必要制御プログラムを記述できたとしても、演算速度の遅いものになったり、制御プログラムの作成(組み合わせ)に大幅に時間や労力を要してしまう。

【0034】本発明はこのような課題に鑑み創案されたもので、ロボット言語又は図形言語により制御可能な制御装置において、システム自体の改造をすることなく、かつ、制御対象に複雑な作業を行なわせる場合にも作業効率を低下させることなく、ユーザが必要な動作を制御対象に行なわせることができるようにした、制御装置を提供することを目的とする。

【0035】

【課題を解決するための手段】このため、請求項1記載の本発明の制御装置は、ロボット言語又は図形言語により記述された制御プログラムに対して標準関数を用いて言語解釈及び演算処理を行なう解釈処理部と、該解釈処理部による該言語解釈及び該演算処理によって得られた指令情報に従って制御対象に動作指令を行なう制御演算部とをそなえた制御装置において、ユーザが該標準関数と別のユーザ関数を登録するユーザ関数登録機能をそなえ、該解釈処理部は、該ユーザ関数に対応した該制御プログラムに対しては該ユーザ関数を用いて言語解釈及び演算処理を行なうことを特徴としている。

【0036】

【発明の実施の形態】以下、図面により、本発明の実施形態について説明する。本実施形態では、本制御装置をロボットの制御に適用した場合を説明する。図1～図6は本発明の一実施形態にかかるロボット制御装置を示すもので、図1はロボット言語の解釈処理のフローを説明する本装置の要部構成図、図2はユーザが作成したユーザ関数を制御装置に登録するためのユーザ関数登録表の一例を示す図、図3はユーザが作成したユーザ関数の演算内容の記述例を示す図、図4はエントリテーブル作成のための記述例を示す図、図5はバッテリバックアップRAMの領域割付例を示す図であって、(a)はユーザ関数登録表とエントリテーブルを別個に作成した場合の領域割付例を示す図、(b)はユーザ関数登録表とエントリテーブルを1つにまとめて作成した場合の領域割付例を示す図、図6はユーザが作成したユーザ関数の使用例を示す図である。

【0037】本実施形態のロボット制御装置(制御装置)のハード構成は、従来技術と同様になっており、図9に示すように、CPU1と、RAM2と、ROM3と、I/Oデバイス4と、バッテリバックアップ付きのRAM5と、それらを互いに繋げる伝送路9とから構成される。そして、本ロボット制御装置には、従来の汎用の制御装置と同様に、ユーザが容易に制御プログラムを記述して用途に応じた動作をロボット(制御対象)に実行させることができるように専用のプログラム言語をそなえた基本ソフトウェアが用意されるとともに、さらに、この基本ソフトウェアにユーザ関数登録機能がそなえられており、ユーザがC言語等の一般プログラム言語を利用して、専用のプログラム言語が標準としてそなえる標準関数とは別のユーザ関数を登録できるようになっている。

【0038】ここで、ユーザ関数登録機能では、ユーザ関数について、ユーザ関数名と、ユーザ関数の演算内容と、ユーザ関数の属性(関数の引数の数及び引数の型等)とを登録するとともに、ユーザ関数を実際に処理するためのアドレスの登録も行なうようになっている。本実施形態では、ユーザの入力が容易となるように、ユーザ関数登録機能を、関数登録と、関数の演算内容の記述と、エントリテーブルの作成との3つに分けて構成している。

【0039】ここで、2つのユーザ関数POW(), KP()の、C言語を利用したユーザ関数登録を例に挙げて、図2～図4によりユーザ登録機能を説明する。図2に示すように、ユーザ関数登録は、ユーザ関数に関数番号(図2中、最も左の欄)を割り付けるとともに、関数の属性として関数の戻り値の型、引数の個数、引数の型を登録するもので、基本ソフトウェアがそなえる表形式によるシステム(登録関数表)によりユーザが容易に入力を行なえるようにしている。図2は、関数番号1に関

数名 $POW()$ を、関数番号2に関数名 $KP()$ を登録した状態を示している。ここで、関数の戻り値の型及び引数の型は、後述のユーザ関数の演算内容の記述と一致するように、 int 、 $float$ といったC言語に準じた型を使用する。

【0040】そして、ユーザ関数の演算内容はC言語により図3のように記述する。図3は $POW()$ の演算内容についての記述であって、べき乗を計算させるものである。また、図示しないが、 $KP()$ の演算内容もC言語により同様に記述する。これらの記述は、C言語のコンパイラにより、制御装置のCPUで動作可能のように実行モジュール(機械語)に変換されるとともにユーザ関数を実際に処理するためのアドレス(エントリアドレス)も割り付けられる。

【0041】ここで、ユーザ関数を実際に使うためには、コンパイラにより割り付けられた各ユーザ関数のエントリアドレスを、基本ソフトウェアが行なう解釈処理で速やかに分かるようにする必要がある。このため、各関数のエントリアドレスを登録するためのエントリテーブルを、例えば、図4に示すC言語の記述によって作成する。ここで、図4の図中に①で示される数はユーザ関数の登録数[本実施例では $POW()$ 及び $KP()$ の2つ]を表し、図4中に②、③で示される数は図2のユーザ関数登録表中の関数番号に一致させて、関数登録表とエントリテーブルとの関連を速やかに確認できるようにしている。以上でユーザ関数登録は終了する。

【0042】ここで、上述のユーザ関数登録により作成した、ユーザ関数登録表、エントリアドレス、ユーザ関数の実行モジュールは、ロボット制御装置のバッテリバックアップ付きのRAM5(図9参照)に図5(a)に示すような割り付け構成でロードして、電源が切れてもロード内容が消えないようにしている。また、ユーザ関数登録表ロード領域及びエントリテーブルのロード領域の先頭アドレスを固定して、ロボット言語解釈部から容易に参照可能なようにしている。

【0043】図6は、上述のようにユーザが定義した関数である $POW()$ 及び $KP()$ の使用例を示している。ステップC10で、ユーザ関数 $POW()$ により変数Bの4乗を計算してこれを変数Aに代入し、さらにステップC20で、ユーザ関数 $KP()$ により、ステップC10で求めた変数Aの値に2.5を乗じたものに、さらに変数Dに4.2を乗じたものを加えて、これを変数Cに代入している。ステップC30では、標準関数である $MOVE$ の引数として、10, 20, 30にステップC20で求めた変数Cを加えたものの3つを取り、制御対象をこの3つの座標へ順次移動させるようにしている。

【0044】本発明の一実施形態にかかる制御装置は、上述のように構成されるので、例えば、図1に示すように、解釈処理を行なうことができる。この解釈処理は、

図10により説明した従来技術の解釈処理に対し、ユーザ関数の解釈処理(ステップD44～ステップD46)を加えたものである。つまり、まず、基本ソフトウェアは制御プログラムを1行ずつ読み込み(ステップD10)、そして、読み込んだプログラム行の字句解析を行なって最小単位の語句(トークン)に分解する(ステップD20)。

【0045】次いで、このトークンを、標準関数記号表及び変数の登録変数記号表と比較して構文解析を行ないトークンが標準関数に合致しているかを調査する。即ち、分解されたトークンが上述の標準関数記号表に登録されているいずれかの標準関数と一致しているか及びその標準関数の属性(標準関数の文法)に合致しているかを調査し、さらに、関数の引数に変数が使われている場合は、その変数が登録変数記号表に登録されている変数(登録変数)のいずれかと一致するとともにその登録変数の属性に合致しているかを調査する(ステップD30)。そしてステップD40へと進む。

【0046】ステップD40では、トークンが標準関数に合致しているかの判定が行なわれ、トークンが標準関数に合致していなければユーザ関数を対象とした構文解析(ステップD44)へ進み、トークンが標準関数に合致していればステップD50へ進んで標準関数に該当する処理を行なうように制御演算部6に指令を出す。ステップD44では、ユーザ関数を対象とした構文解析を行ない、トークンとユーザ関数登録表及び登録変数記号表とを比較してトークンがユーザ関数と合致しているかを調査し、即ち、分解されたトークンが上述のユーザ関数登録表に登録されているいずれかのユーザ関数と一致しているか及びそのユーザ関数の属性(ユーザ関数の文法)に合致しているかを調査し、さらに、ユーザ関数の引数に変数が使用されていれば、その変数が登録変数記号表に登録されている変数(登録変数)のいずれかと一致しているか及びその登録変数の属性に合致しているかを調査する。そして、ステップD46へ進む。

【0047】ステップD46では、トークンがユーザ関数に合致しているかどうかの判定が行なわれ、トークンがユーザ関数に合致していなければ、ステップD100へと進みエラー処理が行なわれて制御プログラムの解釈処理は未完のまま強制終了となり、トークンがユーザ関数に合致していれば、関数番号の情報を伴ってステップD50へ進む。

【0048】ステップD50では、トークンと合致する標準関数またはユーザ関数に該当する処理を行なうように制御演算部6に指令を出してステップD60へと進む。ここで、トークンがユーザ関数に合致している場合はエントリ表からユーザ関数のアドレスの検索を速やかに行なうことができる、つまり、各ユーザ関数に割り当てられた関数番号を介して関数登録表とエントリ表とを関連付けているのでユーザ関数のアドレスの検索を容

易に行なうことができる。

【0049】また、ステップD50、即ち、解釈処理部7から、言語解釈の結果得られた動作命令や移動先の座標などの指令を受けると、制御演算部6は、さらに指令値を計算して、この指令値（動作指令）をI/Oデバイス経由でロボット20の各アクチュエーターに出力する。ステップD60では、ステップD10で読み込んだ制御プログラムの1行が制御プログラムの最終行かどうか、即ち、制御プログラムが終了かどうかの判定が行なわれ、制御プログラムが終了であれば解釈処理は終了するが、制御プログラムが終了でなければ、ステップD10へと戻って、制御プログラムの次の行を読み込み、制御プログラムが終了するまで解釈処理（ステップD10～ステップD60）を繰り返す。

【0050】以上で、基本ソフトウェアによるロボット言語の解釈処理のフローが終了する。本発明の制御装置は、このように、ユーザが目的に応じて一般的なプログラム言語を利用してユーザがユーザ関数を設定できるので以下のような効果がある。ユーザは、目的の作業をロボットにさせるために必要な制御プログラム（必要制御プログラム）をC言語を利用して作成すると、制御装置のCPUで動作可能ように実行モジュール（機械語）に変換されて1つのユーザ関数として登録されるので、処理速度の高速化を図れる。即ち、複数の標準関数を組み合わせたものを解釈処理する場合と、これに相当する1つのユーザ関数を解釈処理する場合とを較べると、前者では標準関数の数だけ図10に示すような解釈処理フローの繰り返しが必要なのに対し、後者では図1に示すような解釈処理を1回行なうだけで良い。

【0051】例えば、掛け算1個にかかる解釈処理時間を約300 μ secに対し、そのうちのCPUでの演算時間は僅か約1 μ secである。ここで、1000個の掛け算が必要なロボットの座標計算を制御装置に行なわせると、従来の制御装置では1000個の掛け算を1個ずつ解釈処理するので約300msecかかるが、本発明にかかる制御装置ではこの1000個の掛け算を機械語に変換して1つのユーザ関数として登録できるので解釈処理は1回で済むため従来の制御装置に較べて、その解釈処理時間は2桁少なくなる。

【0052】さらに、必要制御プログラムをロボット制御装置用に標準で提供される標準関数では記述できない場合でも、豊富な関数をそなえる一般的なプログラム言語であるC言語を利用してユーザ言語を作成することでこの必要制御プログラムを記述することが可能となる。したがって、ユーザは、ロボット制御装置のシステム自体を高価な費用を払ってメーカに改造してもらうことなく、あるいは、標準関数の組み合わせで苦心するという効率の悪い作業を強いられることなく、目的に応じた作業をロボットに行なわせることができる。

【0053】なお、本発明は、上述の実施形態に限定さ

れるものではない。例えば、本実施形態では、C言語によりユーザ関数登録を行なっているが、例えばFORTRANのような他の一般的なプログラム言語を用いてもよい。また、本実施形態では、ユーザ登録機能において、ユーザ関数登録表とエントリテーブルとを別々に構成しているが、これに限定されるものではなく、ユーザ関数登録表とエントリテーブルとを1つにまとめた構成としてもよい。この場合、ロボット制御装置のバッテリーバックアップ付きRAM内の領域構成は、例えば図5(b)に示すようになる。

【0054】さらに、本実施形態では、ユーザ関数登録表は基本ソフトウェアがそなえるシステムにより表形式で入力し、エントリテーブルはC言語により作成しているが、例えば、ユーザ関数登録表をC言語等の一般的なプログラム言語により作成したり、また、エントリテーブルがユーザ関数登録表と連動して自動的に作成されるシステムを基本ソフトウェアに設けてもよい。

【0055】また、本実施形態では、本発明をロボットに適用しているが、本発明は、例えば、プラント等のロボット以外に適用してもよい。これに伴い、ロボット言語以外のプログラム言語、例えばファンクションブロック図(FBD)等の図形言語に適用してもよい。この場合、ユーザは、図11中で1個の四角枠で示される機能要素を一般的なプログラム言語を利用して作成、登録する。

【0056】

【発明の効果】以上詳述したように、本発明の制御装置によれば、一般的なプログラム言語を利用して作成した、ユーザが目的の作業をロボットにさせるために必要な制御プログラム（必要制御プログラム）を、制御装置で動作可能ように実行モジュールに変換して1つのユーザ関数として登録することができるので、処理速度の高速化を図ることができる。さらに、必要制御プログラムをロボット制御装置用に標準で提供される標準関数では記述できない場合でも、豊富な関数をそなえる一般的なプログラム言語を利用して作成したユーザ関数によりこの必要制御プログラムを記述することが可能となる。

【0057】したがって、ユーザは、制御装置のシステム自体を高価な費用を払ってメーカに改造してもらうことなく、あるいは、標準関数の組み合わせで苦心するという効率の悪い作業を強いられることなく、目的に応じた作業をロボットに行なわせることができる。

【図面の簡単な説明】

【図1】本発明の一実施形態にかかるロボット言語の解釈処理のフローを説明する本装置の要部構成図である。

【図2】本発明の一実施形態にかかるユーザ関数登録表の一例を示す図である。

【図3】本発明の一実施形態にかかるユーザ関数の演算内容の記述例を示す図である。

【図4】本発明の一実施形態にかかるエントリテーブル

作成のための記述例を示す図である。

【図5】本発明の一実施形態にかかるバッテリーバックアップRAMの領域割付例を示す図であって、(a)はユーザ関数登録表とエントリテーブルを別個に作成した場合の領域割付例を示す図、(b)はユーザ関数登録表とエントリテーブルを1つにまとめて作成した場合の領域割付例を示す図である。

【図6】本発明の一実施形態にかかるユーザ関数の使用例を示す図である。

【図7】従来の制御装置にかかる塗装ロボットの構成及び塗装ロボットの動作内容を示す図である。

【図8】従来の制御装置にかかるロボット言語（標準関数）による塗装ロボットの制御例を示す図であって、(a)、(b)はそれぞれ図7の動作を塗装ロボットに行なわせるためのロボット言語による制御プログラムを示す図である。

【図9】従来の制御装置にかかるロボット制御装置の内部構成を示す図である。

【図10】従来の制御装置にかかるロボット言語により記述した塗装ロボットの制御プログラムの解釈処理のフローを説明する装置の要部構成図である。

【図11】従来の制御装置にかかるプラント制御装置で用いられるファンクションブロック図(FBD)による制御プログラムの記述例を示す図である。

【符号の説明】

- 1 CPU
- 2 RAM
- 3 ROM
- 4 I/Oデバイス
- 5 バッテリーバックアップ付きのRAM
- 6 制御演算部
- 7 解釈処理部
- 9 伝送路
- 10 ロボット制御装置（制御装置）
- 20 塗装ロボット、ロボット（制御対象）
- 21 塗装スプレー
- 25 移動機構
- 26, 27, 29 傾動関節
- 26a, 27a, 28a, 29a 軸
- 28, 30 回転関節
- 40 作業対象物

【図2】

関数番号	関数名	戻り値の型	引数の個数	引数の型
1	POW	float	2	float, int
2	KP	float	4	float, float, float, float
3				

【図4】

```

/*エントリテーブル記述例*/
extern float POW();
extern float KP();
struct ufunc_t{
    unsigned int no;
    float (*func)();
};
struct ufunc_t funtable[3]{
    ① 2, 0,
    ② 1, POW,
    ③ 2, KP
};
  
```

【図3】

```

/*POW()の演算記述例*/
float POW(a, b)
float a;
int b;
{
    float c;
    int i;
    for(i=0; c=1; i<b; i++){
        c=c*a;
    }
    return c;
}
  
```

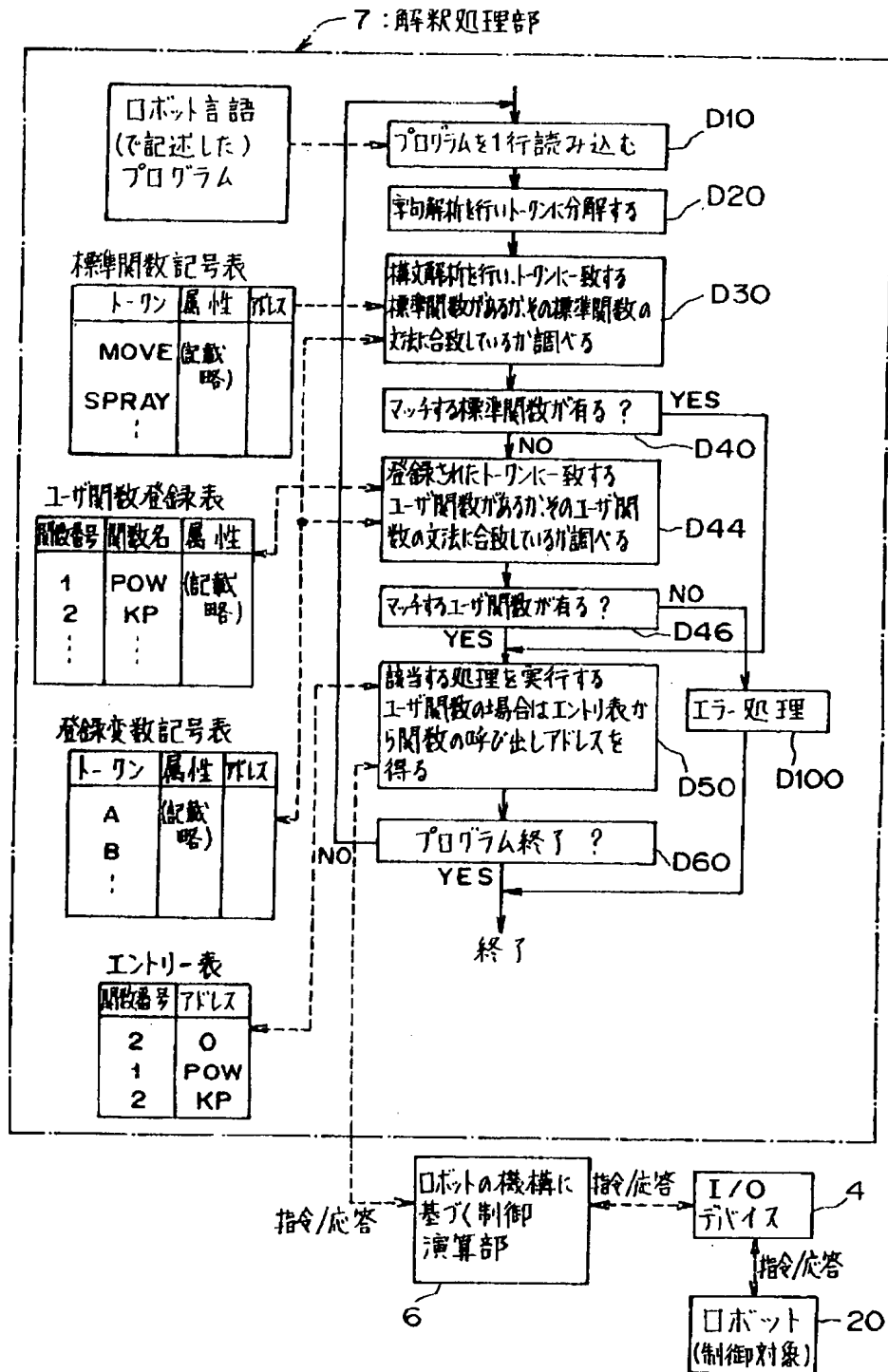
【図6】

```

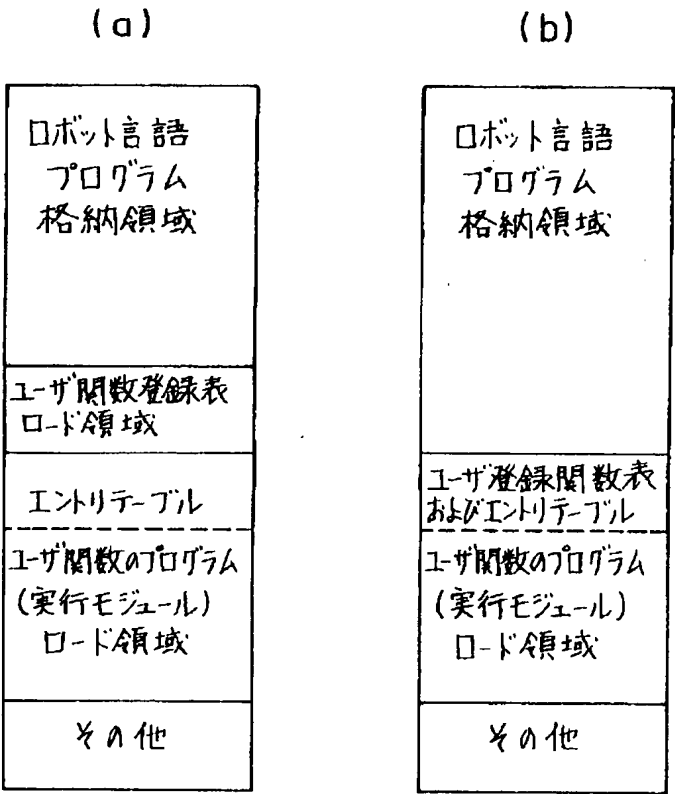
C10 A=POW(B,4)
C20 C=KP(2.5,A,4.2,D)
C30 MOVE IQ,20,30+C
  
```

POW()はBが4乗計算しAに代入
 KP()はC=2.5×A+4.2×D計算
 37B座標値にCを加算しX座標へ移動(MOVE)する

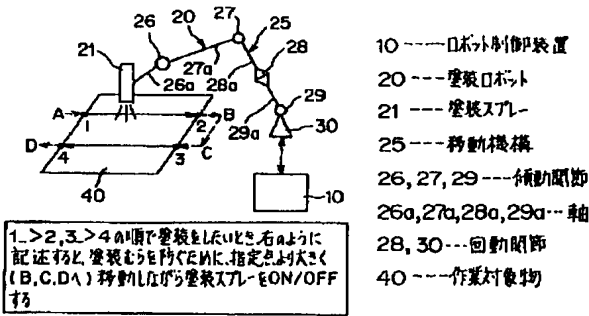
【図1】



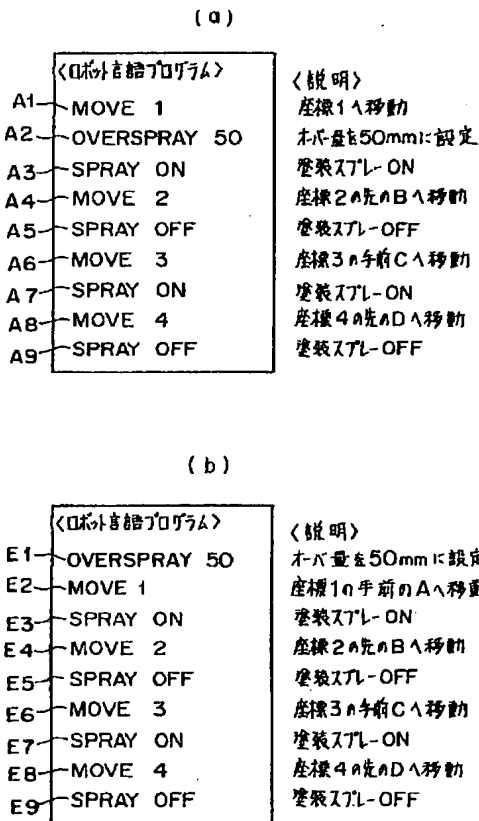
【図 5】



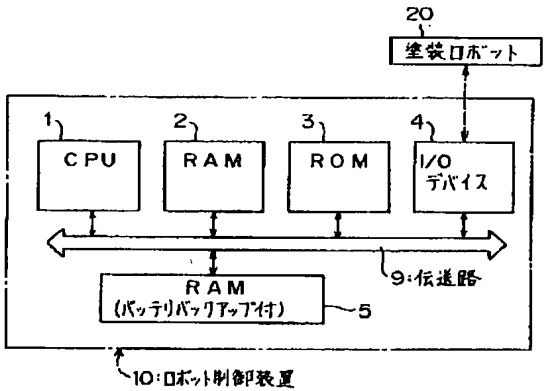
【図 7】



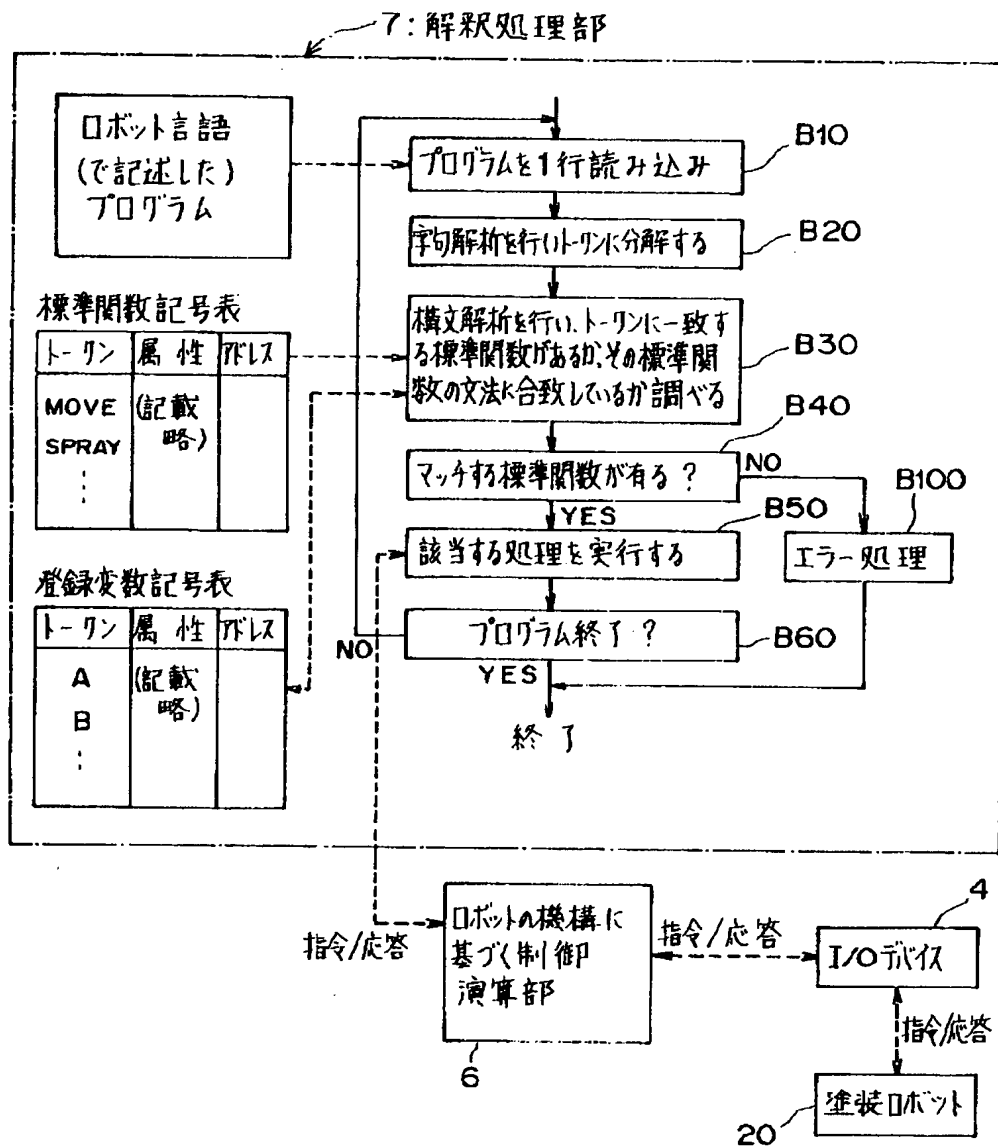
【図 8】



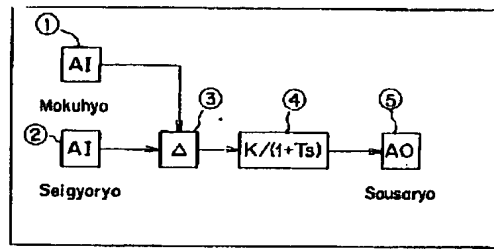
【図 9】



【図10】



【図 1 1】



- AI アナログ入力
 AO アナログ出力
 Δ 減算
 $K/(1+Ts)$ 一次遅れ